ECE 471 – Embedded Systems
Lecture 29

Vince Weaver

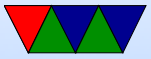
https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

13 November 2023

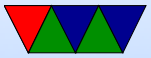
Announcements

- Midterm on Friday
- Don't forget projects
- Don't forget HW#9
- Midterm Friday. Can have one piece of 8.5"x11" paper for notes (single side)
- Project status report due 20th (Monday before Thanksgiving)
  See notes below



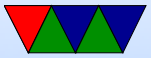
Project Status

- Don't forget project status reports due next Monday (20th)
  - A one-line statement of your project topic
  - A short summary of the progress you've made so far
  - List any parts you need that you don't have yet
  - List if you're willing to present early (Friday the 1st, Monday 4th or Wednesday 6th vs Friday the 8th) (there will be some bonus for presenting early)

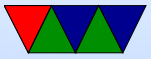


Midterm #2 Preview

- Primarily material since the last midterm
- Booting on the Pi
  - What a bootloader does
  - Why Pi is unusual
- Real Time
  - Definitions
  - Is this hard, soft, firm
- i2c/SPI/1-wire
  - Know the tradeoffs between i2c, SPI, 1-wire

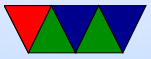


- Be able to follow the C code for them
- Security
  - Buffer overrun, why it is bad
- Coding Practices
  - Be aware of the case studies we suggested
  - Know of some of the recommended ways to write safer C code



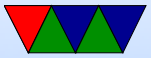
HW#7 Review – Code

- Be sure to `memset(&spi,0,sizeof(struct spi_ioc_transfer));` Not the strings Note saw this in practice. It's tricky to catch as it's a Heisenbug (other code you add can disrupt stack enough to not trigger)
- People managed merging 2 bytes into one OK
- What is the max frequency? Last year someone setting to 500kHz by accident, a few degrees different. Data sheet unclear
- Be sure to check for error on open(), biggest source



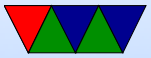
of errors. Linux won't crash, it will happily just report errors that your code is likely ignoring.

- Errors: exiting. Not print plausibly real invalid values. In our case, printing 0V when actually 3.3V not an issue, but imagine if it were 10,000V and you print 0V

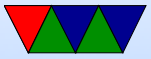


HW#7 Review – Questions

- Disadvantage of SPI?
  More wires, no standard, no errors
- Advantage of SPI?
  Lower Power, Full Duplex, No max speed
- TMP36 on end of cable.
  Voltage Drop, Noise?
  Datasheet has two options, convert to current, or an extra resistor.
- Minimum frequency of 10kHz or results invalid. Maybe

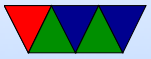


cannot go this fast if bitbanging via GPIO. Also context switch in middle, Linux not realtime?



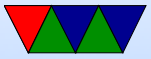
HW#7 Review – Linux "fun"

- Devices here for compatibility with ancient UNIX
  - /dev/null
  - /dev/full
  - /dev/zero, holes in files
  - /dev/random – random numbers (see next slide)
- Windows has ancient compatibility devices too, cause lots of trouble as they are invalid for use in filenames (in any directory, not just /DEV)



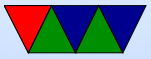
HW#7 Aside – Random Numbers

- Strong random numbers needed for unpredictable crypto, ssh, tls (secure web connections)
- How do you generate random numbers?
- CPUs can provide this. Do you trust them not to have backdoored things? Do the Linux devels?
- Can generate entropy with random I/O, like mouse movements, incoming packets, etc
- Can have pseudo-random number generators (PRNG) Look random, but can predict next result if know seed



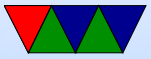
Often used in less critical cases like games and such

Also if you need reproducible random sequences (debugging)



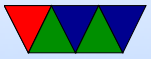
Other I/O You'll find on Embedded Boards

Prioritized for things people might be using for the projects



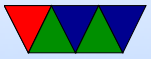
Starting Programs at Boot

- init process starts first
- Traditionally would start various shell scripts under /etc (the name and order of these can vary a lot)
- Possibly with advent of systemd this will change
- Currently you can still put things you want to run at start in `/etc/rc.local`



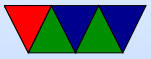
i2c/SPI boards

- If you borrowed a sensor, your best bet is the datasheet
- These are all adafruit breakouts so relatively well documented
- They have sample code, but for arduino or python
- If you run into trouble let me know. Some boards I have sample code for, others might take a bit longer
- Also if device doesn't work let me know. Actually happened last year one of the sensors was broken



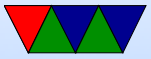
Wii Nunchuck

- Fairly easy to interface
- Put onto i2c bus. Device 0x52
- Send handshake to initialize. Use longer one (0xf0/0x55/0xfb/0x00) not the simpler one you might find(0x40/0x00). This works on generic nunchucks and possibly also disables encryption of results.
- To get values, send 0x00, usleep a certain amount, and read 6 bytes. This includes joy-x, joy-x, accelerometer x/y/z and c and z button data. More info can be found



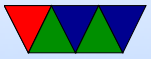
online.

- byte0 = joy-x
- byte1 = joy-y
- byte2 = top 8 bits accelerometer x
- byte3 = top8 acc y
- byte4 = top8 acc z
- byte 5 is bottom 2 z,y,x then button c and z (inverted?)

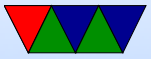


Linux and Keyboard

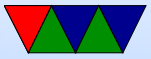
- Old ps/2 keyboard just a matrix of keys, controlled by a small embedded processor.
  Communication via a serial bus. Returns "keycodes" when keypress and release and a few others.
- Many modern keyboards are USB, which requires full USB stack. To get around needing this overhead (for BIOS etc) support bit-bang mode. OS usually has abstraction layer that supports USB keyboards same as old-style



- Linux assumes "CANONICAL" input mode, i.e. like a teletype. One line at a time, blocking input, wait until enter pressed.
- You can set non-CANONICAL mode, async input, and VMIN of 1 to get reasonable input for a game. Arrow keys are reported as escape sequences (ESCAPE-[-A for up, for example).
- Even lower-level you can access "RAW" mode which gives raw keycode events, etc.
- See the `tcgetattr()` and `tcsetattr()` system calls
- There are libraries like ncurses that abstract this a bit.

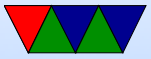


Also GUI and game libraries (SDL).



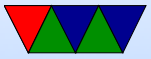
Faking Linux Input Events

- How to insert input events into Linux, i.e. have a software program fake keyboard/mouse/joystick events.
- Linux supports a "uinput" kernel driver that lets you create user input.
- There is some info on a library that makes this easier here: `http://tjjr.fi/sw/libsuinput/`
- It has examples for keyboard and mouse. Joystick should be possible but there's no sample code provided.
- Python wrappers seem to exist too.

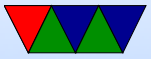


Digital Audio

- How can you generate audio (which is analog waves) with a digital computer?
- One way is PCM, Pulse Code Modulation, i.e. use a DAC.
  - Sample the sound at a frequency (say 44.1kHz), and take amplitude (16-bit audio, 64k possible values)
  - Why 44.1kHz? Nyquist theorem. Twice sample rate to reproduce properly. 22kHz roughly high end of human hearing.

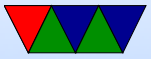


- A WAV file is basically this, has the samples (8 or 16-bit, stereo or mono) sampled at a regular frequency (often 44.1kHz) to play back, write the values to a DAC at the sample rate.



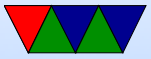
What if no DAC? (Pi has none)

- Can do PWM, Pulse-Width Modulation
- By varying the width of pulses can have the average value equal to an intermediate analog value. For example with duty cycle of 50% average value is 1/2 of Vdd
- Can be "converted" to analog either by a circuit, or just by the inertia of the coil in a speaker.



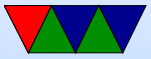
Saving space

- This was more relevant before everyone put all their music in the cloud
- Can be tens of megabytes per song.
- Music can be compressed
- Lossy: MP3, ogg vorbis
- lossless AAC, FLAC



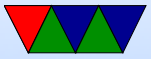
PWM GPIO on Pi

- You can't get good timings w/o real-time OS
- Available on GPIO18 (pin 12)
- Can get 1us timing with PWM in Hardware
  Software: 100us Wiring Pi, less with GPIO interface.
- Which would you want for hard vs soft realtime?
- Other things can do? Beaglebone black as full programmable real-time unit (PRU)
  200MHz 32-bit processor, own instruction set, can control pins and memory, etc.



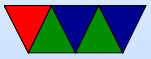
Linux Audio

- In the old days audio used to be just open `/dev/dsp` or `/dev/audio`, then `ioctl()`, `read()`, `write()`
- These days there's ALSA (Advanced Linux Sound Architecture)
  The interface assumes you're using the ALSA library, which is a bit more complicated.
  - Handles things like software mixing (if you want to play two sounds at once)
  - Other issues, like playing sound in background



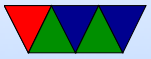
Linux Audio – Libraries

- On top of ALSA is often another abstraction layer
  - pulse-audio – from the same people who made systemd
  - A mixer interface to pick volumes, muting
- More Recently people using pipewire
  aside on DSP to make tiny speakers sound good, intentionally over-drive to sound good but have to model temperature, m1 Linux support has to be careful or you can blow out speakers in software
- JACK interface for low-audio sound between processes



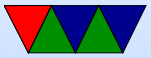
Linux Audio – Programming

- For quick hack can use `system()` to run a command-line audio player like `aplay`
- Better idea might be to use a library such as SDL-mixer which handles all of this in a portable way with a nice library interface.



Pi Limitations

- Pi interface is just a filter on two of the PWM GPIO outputs
- Also can get audio out over HDMI.
- If you want better, can get i2s hat
- Pi lacks a microphone input, so if want audio in on your pi probably need a USB adapter (or one of the microphone boards I have)



i2s

- PWM audio not that great
- i2s lets you send packets of PWM data directly to a DAC
- At least 3 lines. bit clock, word clock (high=right/low=left stereo), data
- Pi support i2s on header 5